The Energy Cost of Secrets in Ad-hoc Networks (Short Paper)

Alireza Hodjat
ahodjat@ee.ucla.edu

Ingrid Verbauwhede
ingrid@ee.ucla.edu

Department of Electrical Engineering
University of California, Los Angeles
Los Angeles CA-90024

Abstract

Energy consumption of cryptographic algorithms and security protocols is a crucial factor in wireless ad-hoc networks. This work explores the energy cost of a key agreement process between two parties of an ad-hoc network using public-key encryption techniques and compares the results with regular networks which use secret-key based key-exchange protocols. Elliptic Curve public-key and Rijndael AES secret-key algorithms are chosen to explore the energy cost of Diffie-Hellman and Kerberos key agreement protocols on a WINS sensor network. The results show that the total energy cost of the Diffie-Hellman key agreement process using Elliptic Curve point-multiplication in an ad-hoc network is between one to two orders of magnitude larger than the key exchange process based on the AES secret-key algorithm in a regular non ad-hoc network.

1. Introduction

Low-energy security is a crucial requirement for wireless networks. The very first step in providing security to a wireless network is the key distribution and key agreement process. Symmetric or asymmetric techniques can be used for key exchange between parties in a wireless network. Protocols based on asymmetric-key encryption algorithms are the main solution for ad-hoc wireless networks because there is no need for a trusted third party in these protocols. On the other hand, symmetric-key encryption algorithms are applicable to the wireless networks with base stations.

The main goal of this paper is to compare the energy cost of security for wireless ad-hoc networks versus non ad-hoc networks which use base stations as their trusted parties. In order to achieve this we explore the energy consumption of the key agreement process between two parties using both symmetric-key and asymmetric-key encryption algorithms.

The platform used for this purpose is the Wireless Integrated Network Sensor. WINS is a StrongARM based wireless sensor network which was developed at Rockwell Scientific [1]. In the first step, symmetric-key and asymmetric-key encryption algorithms are implemented on the WINS nodes. More specifically the Rijndael AES secret-key and Elliptic Curve public-key encryption algorithms are chosen. The energy consumption for these algorithms is measured for different data and key lengths.

In the next step we explore the energy cost of a public-key and a secret-key based key exchange protocol and examine their suitability for low-power wireless networks. In the implementation of the whole security protocol the cost of communication between nodes must be considered as well. Therefore, the whole energy cost consists of the energy consumption of computation and communication. The number of packets exchanged between parties and the energy cost of radio transmission influences the overall communication cost, and the type of encryption algorithm affects the energy cost of computation in each protocol.

The rest of this paper is organized as follows: In section 2 we explain the Rijndael AES secret-key and Elliptic Curve public-key cryptography algorithms and present their energy cost on WINS nodes. Section 3 discusses the energy cost of communication for the WINS nodes. In section 4 the Diffie-Hellman and Kerberos key distribution protocols are presented and the energy cost of implementing these protocols on ad-hoc and non ad-hoc networks is explored. Sections 5 and 6 present our conclusion and future work.

2. Energy results for AES and ECC

In this section we explore the energy cost for the underlying encryption algorithms - Rijndael AES and Elliptic Curve Cryptography (ECC). AES is the secret-key encryption standard most recently accepted by NIST. Other choices for secret-key techniques are DES and RC4. AES was chosen due to the fact that it is more secure and more complex compared to DES and RC4. Moreover, it is more useful in exploring the upper bound of energy for non ad-hoc networks. As for public-key algorithms, the choices are RSA and ECC. ECC has recently gained prominence among public-key algorithms. Furthermore, RSA is more complex and will require much more energy than ECC and hence, is not suitable for ad-hoc networks.

Rijndael Symmetric-key Algorithm

This algorithm was accepted as the Advanced Encryption Standard by NIST [2]. It consists of key scheduling, encryption, and decryption primitives [3]. Key scheduling produces a long array of sub-keys by key expansion and round key selection routines. This long array of sub-keys is

used in each round of encryption or decryption phase to be added to data. Depending on the key and data size, encryption and decryption algorithms are repeated between 10 to 14 rounds. In the encryption primitive, byte substitution transformation, shift row transformation, mix column transformation, and round key addition are performed [3]. In decryption, the steps are the same as those performed in encryption. However, they are called in reverse order and the substitute table, shift indexes, and fixed polynomials used in the decryption steps are the reverse of those used in encryption.

Figure 1 shows the energy consumption of Rijndael AES encryption and decryption algorithms for different data and key lengths. In the original Rijndael algorithm, the input data and key blocks are in the range of 128, 192, or 256 bits, and the output has the corresponding size. The energy consumption for encryption and key scheduling varies from 0.31 *mJoules* to 0.85 *mJoules* depending on the choice of data and key size. For decryption and key scheduling the variation is from 0.36 *mJoules* to 1.01 *mJoules*. The energy consumption of decryption is 30% more than encryption. The main reason for this difference is the number of shifts performed in the shift row routine and the larger $GF(2^8)$ elements used in mix column transformation routine.

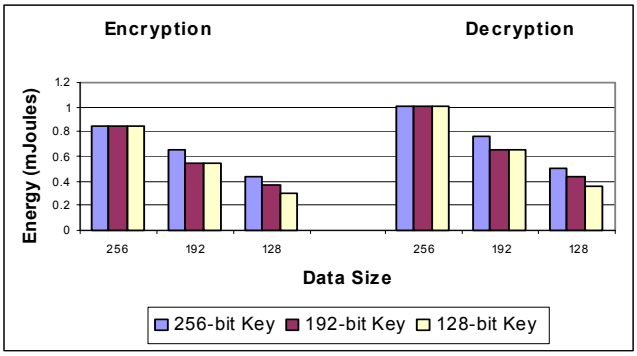

Figure 1: The energy cost of Rijndael AES algorithm on WINS (key scheduling + encryption or decryption)

Elliptic Curve Asymmetric-key Algorithm

This algorithm is based on the IEEE P1363/D1 public-key cryptography standard [4]. We use the efficient double-add-subtract point-multiplication algorithm defined by this standard. Elliptic Curve point-multiplication refers to calculating $k.P$ when k is an integer and P is a point on the Elliptic Curve. The theory of the Elliptic Curve Public-key cryptography is based on the mathematical mapping of an Elliptic Curve on a Galois field. The Elliptic Curve points on $GF(2^n)$ and the point in infinity form a group with a specific addition operation [4]. With this definition, $k.P$ is equivalent to adding P to itself k times by the group operation.

Figures 2, 3, and 4 show the energy consumption of point multiplication for key lengths of 128, 192, and 256 bits, respectively. The execution time and the energy for the

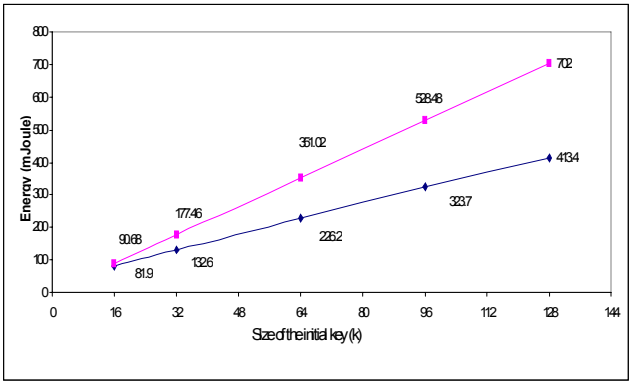

Figure 2: The energy cost of ECC point-multiplication for 128-bit session key

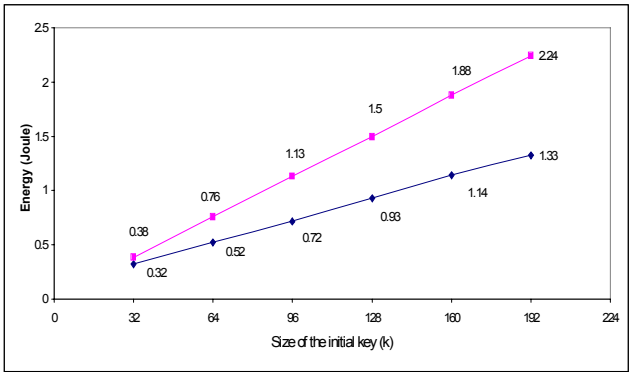

Figure 3: The energy cost of ECC point-multiplication for 192-bit session key

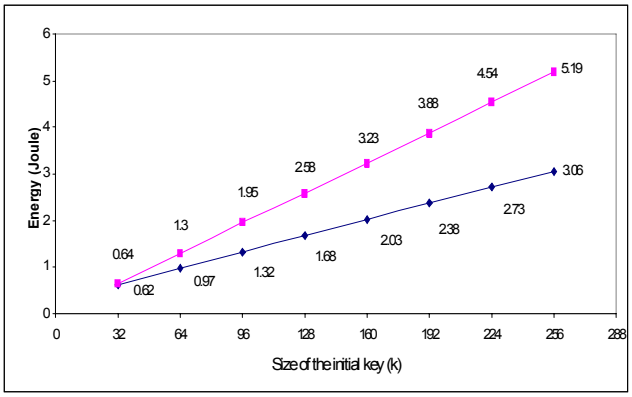

Figure 4: The energy cost of ECC point-multiplication for 256-bit session key

ECC algorithm strictly depend on the bit pattern of the initial random key k. Therefore, the upper bound and lower bound of energy was measured. In order to calculate the lower bound of energy, there should be only doubles in the point-multiplication calculation. Similarly, in order to calculate the upper bound, there should be a full number of doubles and the maximum number of additions and/or subtractions.

In these figures the X-axis shows the size of the initial random key k of point-multiplication. This is identical to

the number of doubles for each experiment. Also, in the case of the upper bound, this axis shows twice the number of adds and/or subtracts. The Y-axis shows the energy consumption in terms of *mJoules* or *Joules*. For instance, if the size of *k* is 64 for 128-bit point multiplication (figure 2), then the energy consumption will be between 226.2 and 351.02 *mJoules* depending on the *0* and *1* pattern in *k*. In this case 64 doubles are performed for the lower limit, and 64 doubles and 32 adds or subtracts are performed for the upper limit.

The results show that the energy consumption of one Elliptic Curve point-multiplication for the 128-bit key varies from 0.082 to 0.702 *Joules*. For the 192-bit key the variation is between 0.32 to 2.24 *Joules*. In the case of the 256-bit key the energy cost range is between 0.62 to 5.19 *Joules*. Also the average energy cost is 0.30, 1.07, and 2.34 *Joules* for the 128, 192, and 256-bit key, respectively. All of the above measurements are done on StrongARM based WINS node running at 133 MHz.

3. Communication costs

[5] presents the power consumption of the radio transmission on WINS nodes. Radio's power consumption varies between 396 to 711 *mWatts* depending on the transmission power level. This corresponds to a consumption of 771 to 1080 *mWatts* for the whole sensor node. The power consumption of the receive mode is 376 *mWatts* for the radio and 751 *mWatts* for the whole node. All these numbers are at a transmission rate of 100 kbits/s.

4. Key-exchange energy consumption

[6] introduces Diffie-Hellman and basic Kerberos as two solutions for the key distribution and key agreement problem. Diffie-Hellman key agreement protocol is implemented using a public-key encryption algorithm. It is applicable to ad-hoc networks due to the fact that it does not require a trusted third party in the key agreement process. Using this protocol, any two nodes in an ad-hoc network can agree on their common session key without any trusted node or secure link. On the other hand, basic Kerberos is a key-distribution protocol that is based on a secret-key encryption technique. It is suitable for non ad-hoc networks with base stations since it requires a trusted party in the key-exchange process. In this experiment on WINS nodes, the Diffie-Hellman protocol is implemented using the Elliptic Curve public-key cryptography technique and the Kerberos protocol is based on the Rijndael AES secret-key cryptography algorithm. In the following paragraphs these protocols are described and their energy consumption is explored. It is assumed that the two nodes performing the key-agreement process are called *Alice* and *Bob*, and if needed, their trusted party is called *Trent*.

In the Diffie-Hellman protocol based on Elliptic Curve point-multiplication, there is a common elliptic curve and a specific point on it that is publicly known. *Alice* and *Bob* each generate a random initial key, called *a* and *b*. They apply point-multiplication on the common elliptic curve point *P*. They exchange their results. Then, they apply point-multiplication on the received data (*a.P* or *b.P)* with their own random initial key again and generate *a.b.P*. This is their common session key. Both can generate *a.b.P* but no one else can generate it without knowing *a* and *b*. It can be shown that it is practically infeasible to generate *a.b.P* from *a.P*, *b.P,* and *P*.

Diffie-Hellman	128-bits	192-bits	256-bits
Four times point multiplication with average energy cost	4×300 mJoules	4×1070 mJoules	4×2340 mJoules
Packet size	128+256	192+256	256+256
Energy cost per packet transmission at maximum power level	4.05 mJoules	4.725 mJoules	10.8 mJoules
Energy cost per packet arrival	2.80 mJoules	3.25 mJoules	7.51 mJoules
Total (mJoules)	1213.7	4296	9378.3

Table 1: Diffie-Hellman based on Elliptic-Curve

In this protocol *Alice* does two elliptic curve point-multiplications (calculating *a.P,* and *a.b.P*), once transmits her results to Bob (*a.P*), and once listens to receive Bob's results (*b.P*). Bob also follows the same process. Therefore, this protocol requires four Elliptic Curve point-multiplications as the computation part, and two data transmissions and two data receptions for the communication part.

Table 1 shows the energy exploration of Diffie-Hellman protocol using Elliptic Curve point-multiplication. A header of 256 bits wide is used for each data transmission and the data is either 128, 192, or 256 bits wide. As it is shown, the average energy consumption of point-multiplication is used in each case based on the results of section 2. Transmission is at the rate of 100 kbits/s and the energy is calculated based on the maximum transmission power level. From section 3 it is observed that the transmission costs maximum 1080 *mWatts* and the reception 751 *mWatts*. As shown in Table 1, the whole energy cost of one Diffie-Hellman key-agreement protocol in an ad-hoc network is 1213.7 *mJoules*, 4296 *mJoules*, and 9378.3 *mJoules* for key lengths of 128, 192, and 256 bits, respectively.

In the basic Kerberos protocol based on the AES secret-key encryption technique *Alice* and *Bob* agree on a common session key with the help of their trusted third party, *Trent*. Here, we focus on the original basic Kerberos key agreement protocol using secret-key encryption technique [6]. We do not discuss the Kerberos network authentication protocol that is used in client/server applications.

Alice and *Bob* want to agree on a common session key. At the beginning, both have a secret key to communicate securely with *Trent*. For key agreement, one of them, suppose *Alice*, sends a message to *Trent* asking for a secret session key. This message includes *Alice's* and *Bob's* identities. *Trent* generates the random session key *K* and

encrypts it along with timestamp *T* and lifetime *L* and *Alice* or *Bob's* identity separately. The message for *Alice* is encrypted with *Alice's* secret-key and *Bob's* identity and vice versa. The results are $E_A(K,B,T,L)$ and $E_B(K,A,T,L)$. These messages are sent to *Alice*. Now *Alice* can decrypt her received message to recover *K*. Then she makes a message including her identity and timestamp *T* and encrypts it with *K* ($E_K(A,T)$). She sends both $E_B(K,A,T,L)$ and $E_K(A,T)$ to *Bob*. Now *Bob* can decrypt the received messages and recover *K*. He also verifies *Alice's* identity and then forms a message with *T+1* and encrypts it with *K* and sends it again to *Alice*. *Alice* decrypts this message and verifies the timestamp *T*. Now *Alice* and *Bob* have a secret common shared key *K*. This process is shown in figure 5.

In this protocol there are four data encryptions on the transmitted data and four decryptions on the received data. The total number of transmissions and receptions is six each. Table 2 shows the upper bound energy consumption of this protocol using the Rijndael AES secret-key algorithm. The packets of size 1 *kbits* include the header, *Alice* and *Bob's* identity, the timestamp, the lifetime, and the generated random session key *K*. *K* can be 128, 192, or 256 bits wide. In order to calculate the upper bound of energy consumption, the maximum power level is considered in transmission. Therefore, the transmission power of 1080 *mWatts* results in energy consumption of 10.8 *mJoules* for the transmission of 1-kbit packet at 100 *kbits/s*. Accordingly, the reception power is 751 *mWatts,* which shows the energy consumption of 7.51 *mJoules*. The encryption and decryption are performed on the block of data with 256-bit length that consumes maximum of 0.85 *mJoules* and 1.01 *mJoules* for encryption and decryption, respectively. Table 2 shows that the energy consumption for Kerberos key-agreement protocol is less than 140 *mJoule*s. This is true for key length of 128, 192, or 256 bits.

5. Conclusion

Our measurements on WINS nodes show an energy cost of maximum 1 *mJoule* for the AES symmetric-key cryptography algorithm. For the Elliptic Curve public-key cryptography algorithm, the average energy consumption of one point multiplication varies between 300 to 2340 *mJoules*. This cost is between two to three orders of magnitude larger than the AES algorithm. This ratio influences the energy cost of the key agreement protocol on an ad-hoc network. Diffie-Hellman key agreement protocol using Elliptic Curve point-multiplication in a typical ad-hoc network costs between 1214 to 9400 *mJoules*. On the other hand it costs less than 140 *mJoules* to exchange keys with an AES-based Kerberos key distribution protocol in non ad-hoc networks. This shows that on WINS nodes, the energy cost of a key agreement process in an ad-hoc network using a public-key encryption technique like ECC is between one to two orders of magnitude larger than key agreement in regular networks with symmetric-key encryption algorithms like AES. Therefore, providing security for ad-hoc wireless networks with public-key algorithms is not only harder than non ad-hoc networks based on secret-key techniques, but also it might cost between 10 to 100 times more energy.

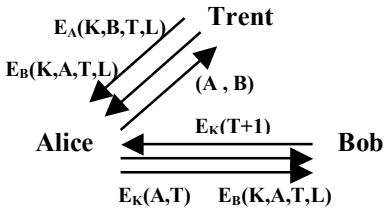

Figure 5: Kerberos key agreement protocol

Kerboros	
Packet Size	1 kbits
Upper bound of energy for encryption of one packet	$4 \times 0.85 = 3.4$ mJoules
Upper bound of energy for decryption of one packet	$4 \times 1.01 = 4.04$ mJoules
Upper bound of energy for encryption in the whole protocol	$4 \times 3.4 = 13.6$ mJoules
Upper bound of energy for decryption in the whole protocol	$4 \times 4.04 = 16.16$ mJoules
Total energy cost of transmission for full size packet	$6 \times 10.8 = 64.8$ mJoules
Total energy cost of reception for full size packet	$6 \times 7.51 = 45.06$ mJoules
Total Energy (mJoules)	139.62

Table 2: Kerberos based on Rijndael AES

6. Future Work

In order to generalize the conclusion, it is helpful to explore the energy cost of the key agreement process for different cases. Other radios, such as Bluetooth or 802.11 can be used. Bluetooth's transmission rate is on the order of a hundred *kbits/s* and its power consumption is in the range of *mWatts*. For 802.11 radio, the transmission rate is in the range of *Mbits/s* and its power consumption is in the order of a hundred *mWatts*. Therefore, the communication cost is more for these radios. Other encryption algorithms like DES, RC4 or RSA can be used in software on processors or in hardware as ASICs. This results in different costs for computation. Exploring the energy cost of the above cases can guide us to find a security model for ad-hoc networks based on the communication/computation trade-off.

References

[1] http://wins.rockwellscientific.com/
[2] http://csrc.nist.gov/encryption/aes
[3] J. Daemen, V. Rijmen, "AES Proposal: Rijndael."
[4] IEEE P1363/D1, Standard Specification for Public-Key Cryptography, November 1999.
[5] Savvides, Park, Sirvastava, "On Modeling Networks of Wireless Microsensors," ACM Sigmetrics 02.
[6] D. R. Stinson, "Cryptography Theory and Practice," First Edition, CRC Press, 1995.